# Tema 9. Bases de Datos Distribuidas (BDD)

***Bases de Datos distribuidas y arquitectura cliente-servidor*** *Elmasri/Navathe 02*

- Conceptos de BDD
- Diseño de BDD
  - Fragmentación
  - Replicación
  - Asignación
- Procesamiento de consultas en BDD
  - Costo de transferir datos y semirreunión
  - Descomposición de actualizaciones y consultas
- BDD y cliente-servidor

*No estudiaremos el punto 24.5, que tiene que ver con administración de BD. El punto 24.7 se ve en el laboratorio*



# Conceptos de BDD

- **Sistema de computación distribuido**: elementos de procesamiento que cooperan en la ejecución de tareas, interconectados por una red de ordenadores.
  - No han de ser homogéneos
  - Dividen un problema en otros más manejables y los resuelven de modo coordinado
- **BD distribuida (BDD):** son varias BD interrelacionadas lógicamente y situadas en diferentes nodos de una red de ordenadores.
- **SGBD distribuido:** el que gestiona BD distribuidas de forma transparente para el usuario (éste ve las BD como si fueran una sola BD centralizada)
- **Ventajas de las BDD:**
  - **Localización transparente** de los datos: las instrucciones no dependen de dónde se ejecutan ni de dónde se sitúan los datos
  - **Transparencia en los nombres**: un objeto se accede por su nombre, sin ambigüedad y sin especificar nada más.
  - **Transparencia de fragmentación**: fragmentación **horizontal** es distribuir una tabla en varios conjuntos de tuplas (cada uno en un ordenador). La **vertical** en distribuir la tabla en conjuntos de atributos. La consulta sobre la tabla se transforma de modo automático en varias consultas sobre sus fragmentos

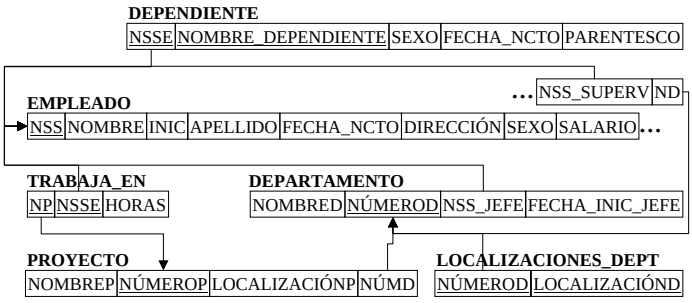


# Conceptos de BDD (2)

- **Ventajas de las BDD** (cont.)**:**
  - **Más fiabilidad y disponibilidad:** Datos y software están en varios ordenadores. Si un ordenador falla los demás pueden seguir funcionando. Los datos y sw del que ha fallado son inaccesibles. Con réplicas (copias) automáticas de datos y sw en varios ordenadores se pueden mejorar estas situaciones.
  - **Mejora del rendimiento**: Situando los datos en el ordenador donde se usan (o uno cercano): BD locales y más pequeñas
  - **Expansión más sencilla**: añadir más datos, más procesadores o aumentar la BD, son tareas más sencillas.
- **Otras funciones de las BDD:**
  - **Seguir la pista a los datos:** fragmentación, réplica
  - **Procesar consultas distribuidas**
  - **Gestionar transacciones distribuidas**
  - **Gestionar datos replicados:** qué copia usar, mantener la consistencia
  - **Recuperar BDD:** de fallos de ordenadores individuales
  - **Seguridad:** privilegios, autorizaciones de acceso
  - **Gestionar el catálogo distribuido:** contiene los metadatos. Debe ser global para toda la BDD o local para cada sitio.



# Esquema de la BD "EMPRESA"



# Estado de la BD "EMPRESA"

**EMPLEADO**

| NOMBRE | INIC | APELLIDO | NSS | FECHA_NCTO | DIRECCIÓN | SEXO | SALARIO | ... |
|---|---|---|---|---|---|---|---|---|
| John | B | Smith | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | H | 30.000 | |
| Franklin | T | Wong | 333445555 | 1955-12-08 | 638 Voss, Houston, TX | H | 40.000 | |
| Alicia | J | Zelaya | 999887777 | 1968-07-19 | 3321 Castle, Spring, TX | M | 25.000 | |
| Jennifer | S | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire, TX | M | 43.000 | |
| Ramesh | K | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble, TX | H | 38.000 | |
| Joyce | A | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX | M | 25.000 | |
| Ahmad | V | Jabbar | 987987987 | 1969-03-29 | 980 Dallas, Houston, TX | H | 25.000 | |
| James | E | Borg | 888665555 | 1937-11-10 | 450 Stone, Houston, TX | H | 55.000 | |

**LOCALIZACIONES_DEPT**

| NÚME-ROD | LOCALIZA-CIÓND |
|---|---|
| 1 | Houston |
| 4 | Stafford |
| 5 | Bellaire |
| 5 | Sugarland |
| 5 | Houston |

**DEPARTAMENTO**

| NOMBRED | NÚME-ROD | NSS_JEFE | FECHA_INIC_JEFE |
|---|---|---|---|
| Investigación | 5 | 333445555 | 1988-05-22 |
| Administración | 4 | 987654321 | 1995-01-01 |
| Dirección | 1 | 888665555 | 1981-06-19 |

| ... NSS_SUPERV | ND |
|---|---|
| 333445555 | 5 |
| 888665555 | 5 |
| 987654321 | 4 |
| 888665555 | 4 |
| 333445555 | 5 |
| 333445555 | 5 |
| 987654321 | 4 |
| nulo | 1 |

**PROYECTO**

| NOMBREP | NÚMEROP | LOCALIZA-CIÓNP | NÚMD |
|---|---|---|---|
| ProductoX | 1 | Bellaire | 5 |
| ProductoY | 2 | Sugarland | 5 |
| ProductoZ | 3 | Houston | 5 |
| Automatización | 10 | Stafford | 4 |
| Reorganización | 20 | Houston | 1 |
| Nuevos beneficios | 30 | Stafford | 4 |

**TRABAJA_EN**

| NSSE | NP | HORAS |
|---|---|---|
| 123456789 | 1 | 32.5 |
| 123456789 | 2 | 7.5 |
| 666884444 | 3 | 40.0 |
| 453453453 | 1 | 20.0 |
| 453453453 | 2 | 20.0 |
| 333445555 | 2 | 10.0 |
| 333445555 | 3 | 10.0 |
| 333445555 | 10 | 10.0 |
| 333445555 | 20 | 10.0 |
| 999887777 | 30 | 30.0 |
| 999887777 | 10 | 10.0 |
| 987987987 | 10 | 35.0 |
| 987987987 | 30 | 5.0 |
| 987654321 | 30 | 20.0 |
| 987654321 | 20 | 15.0 |
| 888665555 | 20 | nulo |

**DEPENDIENTE**

| NSSE | NOMBRE_DEPENDIENTE | SEXO | FECHA_NCTO | PARENTESCO |
|---|---|---|---|---|
| 333445555 | Alice | M | 1986-04-05 | HIJA |
| 333445555 | Theodore | H | 1983-10-25 | HIJO |
| 333445555 | Joy | M | 1958-05-03 | ESPOSA |
| 987654321 | Abner | H | 1942-02-28 | ESPOSO |
| 123456789 | Michael | H | 1988-01-04 | HIJO |
| 123456789 | Alice | M | 1988-12-30 | HIJA |
| 123456789 | Elizabeth | M | 1967-05-05 | ESPOSA |



# Diseño de BDD: fragmentación

- **Directorio (catálogo) global**: contiene la información de fragmentación. Lo utiliza el SBDD.
- **Fragmentar:** decidir dónde situar las partes de la BDD
  - Se puede plantear *top-down* (como aquí) o *bottom-up*
- Idea simple: situar cada tabla en un ordenador distinto
- **Fragmentación horizontal**:
  - Ejemplo: un ordenador por departamento. Cada departamento quiere tener su información en su ordenador. (BD Empresa)

  - Supone dividir las tuplas de cada tabla en 3 trozos. Cada trozo en el ordenador de su departamento.
  - Así, EMPLEADO se divide en: $\sigma_{ND=1}$, $\sigma_{ND=4}$ y $\sigma_{ND=5}$
  - La tabla original se **reconstruye** con UNIÓN de los 3 trozos.
  - Es posible generar fragmentos que compartan tuplas (no disjuntos)
- **Fragmentación horizontal derivada**: la partición de la tabla **primaria** DEPARTAMENTO se aplica a tablas **secundarias**, como EMPLEADO, que la referencian mediante una clave extranjera (ND)



# Diseño de BDD: fragmentación (2)

- **Fragmentación vertical**:
  - De EMPLEADO en información personal y laboral:
    - $\pi_{\text{NOMBRE, INIC, APELLIDO, FECHA_NCTO, DIRECCIÓN, SEXO}}$
    - $\pi_{\text{NSS, SALARIO, NSS_SUPERV, ND}}$
  - Esta división no es apropiada porque no se puede reconstruir la tabla original: es necesario **añadir NSS** a la primera división (clave primaria)
  - Para **reconstruir** la tabla original se usa REUNIÓN EXTERNA COMPLETA (o UNIÓN EXTERNA). *Con REUNIÓN simple, las tuplas de una relación que no se reúnen con ninguna tupla de la otra tabla no aparecen en el resultado.*
  - Podemos generar fragmentos que compartan otros atributos además de la clave (no disjuntos)
- Toda fragmentación debería ser **completa**:
  - Horizontal: todas las tuplas están en algún fragmento
  - Vertical: todo atributo está en algún fragmento
- **Fragmentación mixta**: cuando se aplica fragmentación vertical y horizontal sobre la misma tabla
- **Esquema de fragmentación** es un cjto. de fragmentos que:
  - Fragmentación completa: todos los atributos y tuplas están en algún fragmento
  - Permite **reconstruir** la BD original.
  - Interesante (no necesario) que los fragmentos sean disjuntos



# Diseño de BDD: replicación y asignación

- La **replicación** mejora la disponibilidad de los datos
- Caso extremo: tener **una réplica** de la **BD completa en cada sitio** (ordenador):
  - Ventajas: mejora el rendimiento local y global además de la disponibilidad (con un sitio activo se accede a toda la BD)
  - Inconvenientes: actualizaciones más costosas (se deben realizar en todas las réplicas para mantener la coherencia). El control de concurrencia y recuperación es también más costoso.
- El otro extremo es no tener **ninguna replicación** (salvo las claves primarias en fragmentos verticales).
- Entre ambos extremos: **replicación parcial**. Hay muchas posibilidades.
- **Esquema de replicación**: describe qué se replica

- **Asignación**: dónde se sitúan los fragmentos y réplicas
  - La elección del lugar y el grado de replicación depende de los objetivos de rendimiento y disponibilidad. También del tipo de transacciones y su frecuencia.
  - Encontrar una solución óptima o incluso una buena es un problema complejo

## Ejemplo de diseño de una BDD para EMPRESA

**Enunciado**:

- Tenemos un ordenador para cada departamento. En ***administración*** e ***investigación*** se espera un acceso frecuente a información de empleados y proyectos *del departamento*. En ambos necesitan sólo los atributos *nss, nombre, inic, apellido, salario y nss_superv* del empleado. ***Dirección*** utiliza regularmente toda la información de empleados, proyectos y dependientes.

**Resolución**:

- Se puede almacenar toda la BD en *dirección*. Queda determinar qué fragmentos se replican en *administración* e *investigación.*
- Interesa **fragmentar horizontalmente** DEPARTAMENTO por NÚMEROD y aplicar **fragmentación derivada** en EMPLEADO, PROYECTO y LOCALIZACIONES_DEPT (TRABAJA_EN lo estudiamos aparte)
- Interesa **fragmentar verticalmente** EMPLEADO para los atributos *nss, nombre, inic, apellido, salario, nss_superv y nd.*

(continúa)



## Fragmentos en *administración* e *investigación*

***(falta determinar TRABAJA_EN)***

### *Administración*

**EMPLEADO**

| NOMBRE | INIC | APELLIDO | NSS | SALARIO | NSS_SUPERV | ND |
|---|---|---|---|---|---|---|
| Alicia | J | Zelaya | 999887777 | 25.000 | 987654321 | **4** |
| Jennifer | S | Wallace | 987654321 | 43.000 | 888665555 | **4** |
| Ahmad | V | Jabbar | 987987987 | 25.000 | 987654321 | **4** |

Sólo atributos de interés (vertical)

Sólo tuplas del depto administración (horizontal)

**LOCALIZACIONES_DEPT**

| NÚMEROD | LOCALIZACIÓND |
|---|---|
| **4** | Stafford |

**DEPARTAMENTO**

| NOMBRED | NÚMEROD | NSS_JEFE | FECHA_INIC_JEFE |
|---|---|---|---|
| Administración | **4** | 987654321 | 1995-01-01 |

**PROYECTO**

| NOMBREP | NÚMEROP | LOCALIZACIÓNP | NÚMD |
|---|---|---|---|
| Automatización | 10 | Stafford | **4** |
| Nuevos beneficios | 30 | Stafford | **4** |

**TRABAJA_EN**

| NSSE | NP | HORAS |
|---|---|---|
| … | … | … |

### *Investigación*

**EMPLEADO**

| NOMBRE | INIC | APELLIDO | NSS | SALARIO | NSS_SUPERV | ND |
|---|---|---|---|---|---|---|
| John | B | Smith | 123456789 | 30.000 | 333445555 | **5** |
| Franklin | T | Wong | 333445555 | 40.000 | 888665555 | **5** |
| Ramesh | K | Narayan | 666884444 | 38.000 | 333445555 | **5** |
| Joyce | A | English | 453453453 | 25.000 | 333445555 | **5** |

**LOCALIZACIONES_DEPT**

| NÚMEROD | LOCALIZACIÓND |
|---|---|
| **5** | Bellaire |
| **5** | Sugarland |
| **5** | Houston |

**DEPARTAMENTO**

| NOMBRED | NÚMEROD | NSS_JEFE | FECHA_INIC_JEFE |
|---|---|---|---|
| Investigación | **5** | 333445555 | 1988-05-22 |

**PROYECTO**

| NOMBREP | NÚMEROP | LOCALIZACIÓNP | NÚMD |
|---|---|---|---|
| ProductoX | 1 | Bellaire | **5** |
| ProductoY | 2 | Sugarland | **5** |
| ProductoZ | 3 | Houston | **5** |

**TRABAJA_EN**

| NSSE | NP | HORAS |
|---|---|---|
| … | … | … |



## Ejemplo de diseño de una BDD para EMPRESA (2)

- TRABAJA_EN no tiene número de departamento. Representa una relación M:N entre empleado y proyecto. Podemos elegir entre tomar:
  - Las tuplas de empleados del departamento.
  - Las tuplas de proyectos del departamento
  - Las tuplas de empleados del departamento o de proyectos del departamento
- En nuestro ejemplo decidimos usar la 3ª opción. Por lo tanto puede haber tuplas que figuren tanto en el sitio de *administración* como en *investigación*.
- Esta opción permite hacer la reunión entre empleado y proyecto totalmente en modo local.
- Recordar que en *dirección* hemos dejado la **BD completa**

En negrita los NSSE de empleados del departamento y los NP del depto

### *Administración*

**TRABAJA_EN**

| NSSE | NP | HORAS |
|---|---|---|
| 333445555 | **10** | 10.0 |
| **999887777** | **30** | 30.0 |
| **999887777** | **10** | 10.0 |
| **987987987** | **10** | 35.0 |
| **987987987** | **30** | 5.0 |
| **987654321** | **30** | 20.0 |
| **987654321** | 20 | 15.0 |

### *Investigación*

**TRABAJA_EN**

| NSSE | NP | HORAS |
|---|---|---|
| **123456789** | 1 | 32.5 |
| **123456789** | **2** | 7.5 |
| **666884444** | **3** | 40.0 |
| **453453453** | **1** | 20.0 |
| **453453453** | **2** | 20.0 |
| **333445555** | **2** | 10.0 |
| **333445555** | **3** | 10.0 |
| **333445555** | 10 | 10.0 |
| **333445555** | 20 | 10.0 |

repetido



## Procesamiento de consultas en BDD: *coste de transferir datos*

- Algoritmos de optimización de consultas en SGBDD: *reducir la cantidad de datos a transferir*

### *Sitio 1*

**EMPLEADO**

| NSS | NOMBRE | INIC | APELLIDO | FECHA_NCTO | DIRECCIÓN | … | SEXO | SALARIO | NSS_SUPERV | ND |
|---|---|---|---|---|---|---|---|---|---|---|

9 bytes | 15 bytes | 15 bytes | … | 4 bytes

10.000 registros 100 bytes/fila

### *Sitio 2*

**DEPARTAMENTO**

| NOMBRED | NÚMEROD | NSS_JEFE | FECHA_INIC_JEFE |
|---|---|---|---|

10 bytes | 4 bytes | 9 bytes

100 registros 35 bytes/fila

**SELECT** Nombre, Apellido, NombreD
**FROM** Empleado **inner join** Departamento **on** ND=NumeroD

- Resultado 10.000 filas (una por empleado, si todo empleado tiene ND) de 40 bytes/fila
- La consulta se solicita en **Sitio 3.** Hay tres estrategias:
  - Transferir Empleado y Departamento a Sitio 3. Hacer allí la reunión: 10.000*100 + 100*35 = **1.003.500 bytes transferidos**
  - Transferir Empleado a Sitio 2. Hacer allí la reunión y enviar el resultado a Sitio 3: 10.000*100 + 10.000*40 = **1.400.000 bytes transferidos**
  - Transferir Departamento a Sitio 1. Hacer allí la reunión y enviar el resultado a Sitio 3: 100*35 + 10.000*40 = **403.500 bytes transferidos → mejor opción**

## Procesamiento de consultas en BDD: *coste de transferir datos (2)*

- La misma consulta se solicita en **Sitio 2.** Dos estrategias:
  - Transferir Empleado a Sitio 2 y hacer allí la reunión: 10.000*100 = **1.000.000 bytes transferidos**
  - Transferir Departamento a Sitio 1. Hacer allí la reunión y enviar el resultado a Sitio 2: 10.000*40 + 100*35 = **403.500 bytes transferidos → mejor opción**

- Operación de **semirreunión ( |x ):**
  - Es otra estrategia que a veces mejora los resultados
  - Se basa en transferir solamente las tuplas y atributos estrictamente necesarios

- En el caso de la consulta anterior, solicitada en Sitio 2, una estrategia con **semirreunión** puede ser:
  - Transferir **R1**=$\pi_{NumeroD}$ (Departamento) a Sitio 1: 4*100=**400 bytes**
  - R1 se reúne con Empleado en Sitio 1. Transferir a Sitio 2 **R2**=$\pi_{Nombre,\ Apellido,\ ND}$ (R1|x| Empleado): 34*10.000 = **340.000 bytes**
  - R2 se reúne con Departamento en Sitio 2 para obtener el resultado de la consulta.
  - Con esta estrategia se transfieren **340.400 bytes → mejor opción que las anteriores**

- Ejercicio. Hacer el mismo estudio para la siguiente consulta:
  **SELECT** Nombre, Apellido, NombreD
  **FROM** Empleado **inner join** Departamento **on** NSS=NSS_Jefe



## Descomposición de actualizaciones y consultas

- SGBD sin **transparencia de distribución**: hay que indicar el sitio y la tabla sobre la que se realiza la consulta.
- SGBD sin **transparencia de replicación**: hay que mantener a mano la consistencia de los datos
- SGBD con **transparencia de distribución, replicación y fragmentación**:
  - La consulta o actualización se expresan como si se tratase de un SGBD centralizado
  - El SGBD se encarga de descomponer y dirigir a los fragmentos adecuados



## BDD y cliente-servidor: *arquitectura de 2 niveles*

- Los SGBD totalmente distribuidos (transparentes) aun no son viables comercialmente
- En su lugar se han creado sistemas basados en cliente-servidor
- La forma habitual de dividir la funcionalidad del SGBD entre cliente y servidor ha sido la **arquitectura de 2 niveles**:
  - **Servidor** (o servidor SQL): donde se sitúa el SGBD. Una BDD se situaría en varios servidores.
  - **Clientes**:
    - Envían consultas/actualizaciones a servidores
    - Tienen interfaces SQL, de usuario y funciones de interfaz del lenguaje de programación
    - Consultan en el diccionario de datos la información sobre la distribución de la BD entre los servidores. Tienen módulos que descomponen consultas globales en varias locales a cada servidor
- Interacción cliente-servidor (***arquitectura de 2 niveles***):
  - El cliente analiza la consulta del usuario. La descompone en varias subconsultas y envía cada una a un servidor. (También puede hacerlo el usuario a mano)
  - Cada servidor ejecuta su subconsulta y devuelve el resultado al cliente
  - El cliente combina los resultados recibidos y muestra al usuario el resultado de su consulta.
- En este enfoque al servidor se le llama máquina *back-end* (o subyacente) y al cliente máquina *front-end* (de la parte visible).
- Al servidor también se le llama *servidor de transacciones* y *procesador de BD* y al cliente *procesador de aplicaciones*



## BDD y cliente-servidor: *arquitectura de 3 niveles*

- Actualmente es más común utilizar una arquitectura en 3 niveles, sobre todo para aplicaciones web.
- Las 3 capas son:
  - **Cliente** (Presentación):
    - Es la interfaz (interfaces web, formularios, …)
    - Suelen usar navegadores web y lenguajes como HTML, JavaScript, PERL, …
    - Gestiona las entradas, salidas y la navegación con páginas web estáticas o, cuando accede a BD, con páginas dinámicas (ASP, JSP, …)
  - **Servidor de aplicaciones (SA)** (lógica de negocio):
    - Incluye, por ejemplo, consultas basadas en datos introducidos por el usuario, o resultados de consultas a los que da formato y envía para su presentación
    - Puede incluir otro tipo de funcionalidad como comprobaciones de seguridad o de la identidad
    - Puede acceder a varias BD conectándose mediante ODBC, JDBC u otras técnicas
  - **Servidor de BD (SBD)**:
    - Procesa consultas y actualizaciones solicitadas por la capa de **aplicación**
    - Puede devolver los resultados en formato XML
- La división de funcionalidad entre las 3 capas puede variar.

## BDD y cliente-servidor:
### *arquitectura de 3 niveles (2)*

- El **servidor de aplicaciones** (**SA**) también:
  - Tiene acceso a *diccionarios de datos* para consultar cómo se distribuyen las BDD entre los **servidores de BD (SBD)**
  - Puede incluir módulos para descomponer una consulta en varias subconsultas locales a cada **SBD**
- La interacción entre **SA** y **SBD** puede ser así:
  - El **SA** construye una consulta utilizando datos tomados por el cliente. Descompone la consulta en varias subconsultas, cada una de ellas local a un **SBD**, y las envía a sus correspondientes **SBD**.
  - Cada **SBD** procesa sus consultas y envía los resultados al **SA** que las solicitó. Cada vez es más frecuente utilizar el formato XML para devolver los resultados.
  - El **SA** combina los resultados para obtener el resultado de la consulta original. Dota al resultado de un formato, como HTML y lo envía al cliente para su presentación.
- El **SA** es responsable de (en arquitectura de 2 niveles lo es el cliente):
  - Generar un *plan de ejecución* distribuido y supervisar su ejecución.
  - Garantizar la *consistencia* de las réplicas de datos.
  - Asegurar la *atomicidad* de las transacciones globales (que no se puedan ejecutar "a medias", o sea que bien se ejecuta toda la transacción o no se ejecuta nada)



# Ejercicios



## Ejercicio de diseño de BDD: Sociedad médica

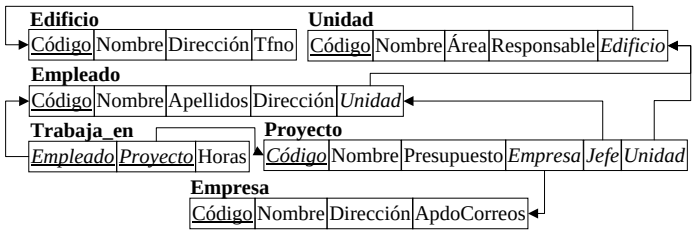
- Cuenta con una oficina central y 3 centros médicos. Cada centro médico atiende ciertas especialidades. Una especialidad se puede atender en varios centros. Todo centro médico tiene al menos una especialidad.
- Actualmente utilizan la siguiente BD relacional centralizada:

- Diseñar esquemas de *fragmentación*, *replicación* y *asignación* de una BDD que tenga la mayor autonomía local, sabiendo que en los centros se necesita:
  - En la oficina central (Centro.Código=0) la información para realizar las nóminas de todos los empleados.
  - En el resto de centros, los horarios de sus consultas y la información de su personal y de sus especialidades.



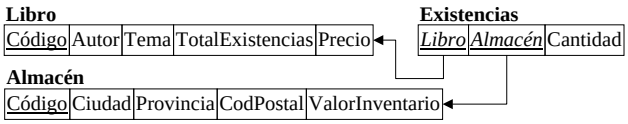
## Ejercicio de diseño de BDD: Servicios informáticos

- Se divide en cuatro *edificios*, cada uno con su servidor de BD, donde hay una o varias *unidades de desarrollo*. Cada una se aloja en un solo edificio. Los *empleados* y *proyectos* están asignados a una *unidad de desarrollo*, aunque un *empleado* puede trabajar en un *proyecto* que no sea de su *unidad*.
- Actualmente utilizan la siguiente BD relacional centralizada:




- Diseñar esquemas de *fragmentación*, *replicación* y *asignación* de una BDD sabiendo que :
  - Cada unidad gestiona las nóminas de sus empleados y necesita la información de los proyectos que realiza.
  - La unidad de *Recursos humanos* (edificio 0) realiza la facturación y usa todos los datos de las empresas. El resto de unidades sólo necesita el código y nombre de las empresas con las que trabaja.
  - Hay proyectos internos, y por tanto sin factura, que tienen valor nulo en su campo Empresa.

# Ejercicio de consultas en BDD: Libros




- Esquema de fragmentación:
  - Libro_a: $\sigma_{Precio<=20€}$(Libro)
  - Libro_b: $\sigma_{Precio>20€ \text{ and } Precio<=50€}$(Libro)
  - Libro_c: $\sigma_{Precio>50€ \text{ and } Precio<=100€}$(Libro)
  - Libro_d: $\sigma_{Precio>100€}$(Libro)
  - Almacén_1: $\sigma_{CodPostal<=3500}$(Almacén)
  - Almacén_2: $\sigma_{CodPostal>3500 \text{ and } CodPostal<=70000}$(Almacén)
  - Almacén_3: $\sigma_{CodPostal>70000}$(Almacén)
  - Existencias_i: Existencias $|\times_{Almacén=Código}$ Almacén_i
- Esquema de replicación y asignación:

| | **Servidor 1** | **Servidor 2** | **Servidor 3** |
|---|---|---|---|
| **Libro** | Libro_a<br>Libro_d | Libro_a<br>Libro_b | Libro_a, Libro_b<br>Libro_c, Libro_d |
| **Almacén** | Almacén_1 | Almacén_2 | Almacén_3 |
| **Existencias** | Existencias_1 | Existencias_2 | Existencias_3 |

- Qué subconsultas genera la ejecución de la siguiente consulta en el servidor 1:

  **select** Código, TotalExistencias
  **from** Libro
  **where** Precio>15 and Precio<55
- Qué modificaciones genera la actualización del precio del libro con código 1234 de 45€ a 55€ en el servidor 2
- Escribir un ejemplo de consulta que ejecutada en el servidor 3 genere una subconsulta en el servidor 2